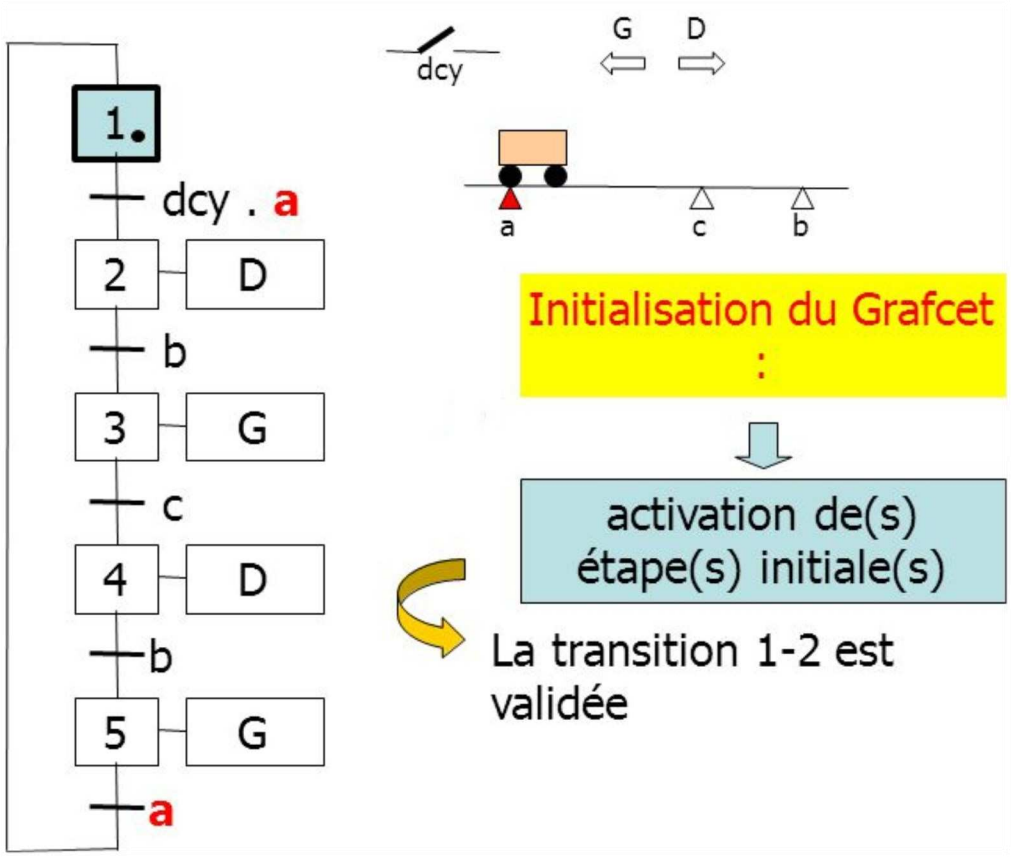
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
ISET — Nabeul
Département Génie mécanique



COURS

AUTOMATISMES INDUSTRIELS

2ÈME ANNÉE GÉNIE MÉCANIQUE

Moez AYADI

A.U. : 2016 / 2017

Avant-propos

Ce cours a été réalisé pour servir comme support aux étudiants de l'ISET de Nabeul. Il ne prétende en aucun cas regrouper toute la science de l'automatisme industriel, mais se place plutôt comme un document permettant une initiation à cette science en présentant quelques-unes de ses multiples facettes.

Son élaboration est le fruit d'un travail minutieux de recueil, de lecture et de synthèse de diverses sources documentaires : livres, cours et sites internet.

L'auteur tient à mentionner que la plupart des figures et illustrations présentes dans ce cours sont à la propriété des détendeurs du droit d'auteur et que leur reproduction dans ce cours est uniquement pour une utilisation pédagogique.

SOMMAIRE

PARTIE A : SYSTEME COMBINATOIRE 1

CHAPITRE I : Système de numération et codage de l'information 2

I. Systèmes de numérations 2
1. Système décimal 2
2. Système binaire 2
2.1. Conversion binaire-décimal 2
2.2. Conversion décimal-binaire 3
3. Système octal 3
3.1. Conversion octal-décimal 3
3.2. Conversion décimal-octal 3
4. Système hexadécimal 4
4.1. Conversion hexadécimal-décimal 4
4.2. Conversion décimal- hexadécimal 4
4.3. Conversion hexadécimal-octal 5
II. Notion de codage 5

CHAPITRE II : Algèbre de Boole et porte logique 7

I. Introduction 7
II. Variables et fonctions logiques 7
1. Variables logiques 7
2. Fonctions logiques 8
3. Eléments de base 8
3.1. Variables d'entrées 8
3.2. Variables de sorties 8
III. Les opérations fondamentales de l'algèbre de Boole et les propriétés associées 8

1. Structure de l'algèbre de Boole 8
2. Propriétés 9
2.1. Sur une seule variable 9
2.2. Sur plusieurs variables 9
IV. Matérialisation des opérateurs logiques 10
1. Porte NON 10
2. Porte OU 10
3. Porte ET 10
4. Porte NAND 10
5. Porte NOR 10
6. Porte XOR 11
CHAPITRE III : Recherche et simplification des fonctions logiques combinatoires 12
I. Introduction 12
II. Fonctions logiques décrites par une table de vérité 12
1. Fonction complétement définie 12
2. Fonction incomplètement définie 12
3. Recherche d'une fonction logique à partir de sa table de vérité 13
III. Simplification des fonctions logiques 13
1. Simplification algébrique 13
2. Simplification graphique par le tableau de KARNAUGH 14
2.1. Règles de simplification 14
2.2. Diagramme de KARNAUGH 14
2.3. Simplification des expressions logiques à l'aide du tableau de KARNAUGH 15
PARTIE B : SYSTEME SEQUENTEIL 17
CHAPITRE I : LE GRAFCET 18
I. Eléments de base 18
1. Etapes et actions associées 18

2. Transitions et réceptivités associées 19
II. Application 19
- Poste de perçage 19
- Travail demandé 20
- Réponse 20
III. Structure de base d'un GRAFCET 21
1. Structure à séquence unique 21
2. Structure à séquence simultanées 22
3. Structure à sélection de séquence 22
IV. Liaison entre GRAFCETS 23
V. Mise en équation d'un GRAFCET 23
1. Règle générale 23
VI. Les notions avancées 24
1. Application 24
2. Structures hiérarchisées d'un grafcet 25
2.1. Structure d'un sous-grafcet 25
2.2. Structure d'un grafcet de tache 26
2.3. Structure de l'expansion en macro-étape 26

CHAPITRE I : LE GEMMA 28

I. Introduction 28
II. Les états 28
1. Les états « F » : Procédures de fonctionnement 28
2. Les états « A » : Procédures d'd'arrêts 29
3. Les états « D » : Procédures en défaillances 30
III. Sélection des modes de marche et d'arrêt 32
IV. Conditions d'évolution entre modes de marches et d'arrêts 32

CHAPITRE I : Les automates programmables industriels (API) 34

I. Introduction 34
II. Structure générale d'une installation automatisée 34
III. Structure générale des API 35

IV. Critères de choix d'un API 35
V. Exemples d'API 36
VI. Programmation de l'automate siemens 36
1. Affectation des bornes du S7-200 (CPU 221) 36
2. Types de variables 36
3. Jeu d'instructions 37
4. Sécurité 38
5. Exemple de programmation LADDER 39
6. Application 40

PARTIE A

SYSTEME COMBINATOIRE

Chapitre 1 : Système de numération et codage de l'information

I.Systèmes de numérations :

L'ensemble des outils informatiques sont basés sur les mêmes principes de calcul (loi de tout ou rien). Les calculs habituels sont effectués dans le système de numération décimal, par contre le calculateur électronique ne peut pas utiliser ce système car le circuit électronique ne permet pas de distinguer 10 états. Le système de numération binaire ne comportera que 2 états 0 ou 1.

1. Système décimal: (base 10: 10 éléments de 0 à 9)

- Exemples:

$6713 = 6.10^3 + 7.10^2 + 1.10^1 + 3.10^0$

$852,39 = 8.10^2 + 5.10^1 + 2.10^0 + 3.10^{-1} + 9.10^{-2}$

2. Système binaire: (base 2: 2 éléments 0 et 1 => 2 bits)

1.1-Conversion binaire-décimal:

- Exemple n°1: $101011_{(2)}$

On fait la somme des pondérations (pondérations = bit ×puissance)

On aura : $1.2^5+0.2^4+1.2^3+0.2^2+1.2^1+1.2^0 = 32+8+2+1=43_{(10)}$

- Exemple n°2: 101,101(2)

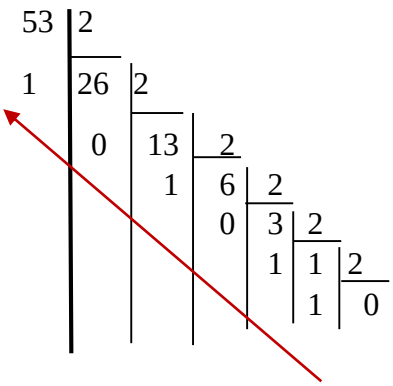
Somme des pondérations : $1.2^2+0.2^1+1.2^0+1.2^{-1}+0.2^{-2}+1.2^{-3} = 4+1+0,5+0.125 =5.625_{(10)}$

2.2- Conversion décimal-binaire:

Exemple n°1:

$53_{(10)} = ?_{(2)}$

On fait la division successive par la base.



Les restes de cette division donne le nombre dans la base 2, soit :

$53_{(10)} = 110101_{(2)}$

3/ Système octal: (base 8: 8 éléments de 0 à 7)

3.1-Conversion octal-décimal:

- Exemple: $326_{(8)} = 3.8^2+2.8^1+6.8^0 = 192+16+6 = 214_{(10)}$

3.2-Conversion décimal-octal:

On a 2 méthodes: « passage par la base 2 » ou « divisions successives par 8 »

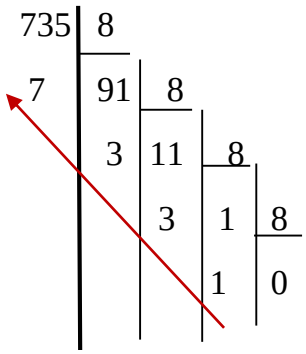
Octal	Binaire
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

- Passage par la base 2 :

  - Exemple: $735_{(10)} = ?_{(8)}$

$735_{(10)} = 1\ 011\ 011\ 111_{(2)} = 1337_{(8)}$

- divisions successives par 8



4/ Système hexadécimal: (base 16: 16 éléments, 0..9,A,B,C,D,E et F)

4.1- Conversion hexadécimal-décimal:

- Exemple: $C3A5_{(16)} = ?_{(10)}$

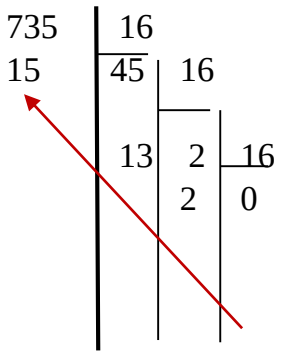
$3A5_{(16)} = 3.16^2+10.16^1+5.16^0 = 768+ 160+5 = 933_{(10)}$

4.2- Conversion décimal-hexadécimal:

On a 2 méthodes: « passage par la base 2 » ou « divisions successives par 16 »

Base 10	Base 16	Base 2
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

- passage par la base 2
  - Exemple: $735_{(10)} = ?_{(16)}$
  $735_{(10)} = 10\ 1101\ 1111_{(2)} = 2DF_{(16)}$
- division successive par 16



$735_{(10)} = 2DF_{(16)}$

4.3- Conversion hexadécimal-octal:

- Exemple: $9D1_{(16)} = ?_{(8)}$

$9D1_{(16)} = 1001\ 1101\ 0001_{(2)}$

$= 100\ 111\ 010\ 001_{(2)}$

$= 4721_{(8)}$

II- Notions de codage:

Un dispositif logique ou numérique est destiné à manipuler des informations diverses qui doivent être traduites par un ensemble de 0 et 1, obtenu suivant une loi de correspondance préétablie: c'est l'opération de codage de l'information.

Codage (loi de correspondance)

Information ⟶ **{Configuration binaire}**

-Exemples de codes :

* *Code ASCII:* chaque touche du clavier est codée sur 8 bits, donc on peut coder 256 caractères. Exemple: Touche 'A' ==> code ASCII « 01000001 »

Upper (left-hand) bits:		0000	0001	0010	0011	0100	0101	0110	0111
Group marker bits:		^^	^^	^^	^^	^^	^^	^^	^^
Group bit pattern:		x00xxxxx		x01xxxxx		x10xxxxx		x11xxxxx	
Group function:		Control		Pnc/Sym/Num		Uppercase		Lowercase	
Lower (right-hand) bits:	0000	nul	dle	spc	0	@	P	`	p
	0001	soh	dc1	!	1	A	Q	a	q
	0010	stx	dc2	"	2	B	R	b	r
	0011	etx	dc3	#	3	C	S	c	s
	0100	eot	dc4	$	4	D	T	d	t
	0101	enq	nak	%	5	E	U	e	u
	0110	ack	syn	&	6	F	V	f	v
	0111	bel	etb	'	7	G	W	g	w
	1000	bs	can	(	8	H	X	h	x
	1001	ht	em	)	9	I	Y	i	y
	1010	nl	sub	*	:	J	Z	j	z
	1011	vt	esc	+	;	K	[	k	{
	1100	np	fs	,	<	L	\	l	|
	1101	cr	gs	-	=	M	]	m	}
	1110	so	rs	.	>	N	^	n	~
	1111	si	us	/	?	O	_	o	del

Comment trouver la valeur de la lettre A ?

En haut 0100 donne une valeur de 4 et à gauche 0001 donne une valeur 1 donc la valeur de la lettre A est égale à 41

Tableau du code ASCII

* *Code DCB* (Décimal Codée en Binaire): utilisé uniquement pour les chiffres décimaux. Ce code est obtenu en remplaçant individuellement chacun des chiffres du nombre à représenter par son équivalent binaire pur.

Exemple : $7592_{(10)}$= 0111 0101 1001 $0010_{(DCB)}$

Chapitre 2 : Algèbre de Boole et portes logiques

I- Introduction:

Historique:

George Boole (1815-1864) est un mathématicien autodidacte anglais qui voulait faire un lien entre la logique (étude de la validité du raisonnement) et la représentation symbolique utilisée en mathématique.

Claude Shannon montre que l'algèbre de Boole peut être utilisée pour optimiser les circuits. Cette nouvelle avenue de recherche va ouvrir la voie à l'ère numérique.

En utilisant l'algèbre de Boole avec le système binaire, on peut concevoir des circuits capables d'effectuer des opérations arithmétiques et logiques.

Définition:

L'algèbre de Boole est l'outil mathématique qui permet d'établir la relation entre les sorties et les entrées d'un système logique (synthèse du système). Réciproquement, cet outil nous permet de déterminer les règles de fonctionnement d'un système logique existant (analyse du système).

* Les opérateurs élémentaires de l'algèbre sont matérialisés par des systèmes physiques: optiques, pneumatiques ou électriques.

En technologie électronique:

- les variables logiques sont généralement des signaux « bi-tension »,
- les opérateurs logiques sont des circuits électroniques appelés « portes logiques ».

II- variables et fonctions logiques

1 / variables logiques :

Une variable ne peut prendre que 2 valeurs notées 0 et 1, qui représentent l'état d'un système bistable générateur de la variable physique.

2 / fonctions logiques :

C'est l'expression de variables et d'opérateurs logiques. f (a,b,c….,n)

3/ Éléments de base

3.1-Variables d'entrée

– Les variables d'entrée sont celles sur lesquelles on peut agir directement. Ce sont des variables logiques indépendantes.

3.2-Variable de sortie

– Variable contenant l'état de la fonction après l'évaluation des opérateurs logiques sur les variables d'entrée.

III- les opérations fondamentales de l'algèbre de Boole et les propriétés associées.

1/ Structure d'algèbre de Boole

- 2 lois de composition interne (OU, ET)
- 1 application unaire (NON)

-Lois de Composition Interne :

- ET, OU Somme (OU, Réunion) s = a + b = a OU b = a **or** b

Nb: a+b se lit « a OU b » pas « a PLUS b »

S est vrai si a OU b est vrai.

a	b	S
0	0	0
0	1	1
1	0	1
1	1	1

- produit (ET, intersection) s = a . b = ab= a ET b = a **and** b

a	b	s
0	0	0
0	1	0
1	0	0
1	1	1

-Application unaire :

Not (complementation, inversion) s =$\bar{a}$ = NON a = **not** (a)

2/ propriétés:

2.1- sur une seule variable :

Opérateur OU	Opérateur ET
A+A = A	A.A = A
A+1 = 1	A.1 = A
A+0 = A	A.0 = 0
A+ A = 1	A. A = 0
Elément neutre = 0	Elément neutre = 1

2.2- sur plusieurs variables:

- **Commutativité**

a+b = b+a

a.b = b.a

- **Associativité**

a+ (b+c) = (a+b) +c

a. (b.c) = (a.b).c

- **Distributivité**

a. (b+c) = a.b+a.c

a+ (b.c) = (a+b). (a+c)

- **Absorption**

a+a.b = a

a. (a+b) = a

- Théorème de Morgan

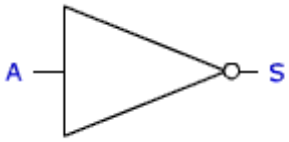
$$\overline{a+b} = \bar{a}.\bar{b}$$

$$\overline{a.b} = \bar{a} + \bar{b}$$

IV-Matérialisation des opérateurs logiques :

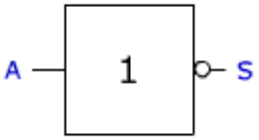
En électronique digitale, les opérations logiques sont effectuées par des portes logiques. Ce sont des circuits qui combinent les signaux logiques présentés à leurs entrées sous forme de tensions. On aura par exemple 5V pour représenter l'état logique 1 et 0V pour représenter l'état 0.

1/ porte NON

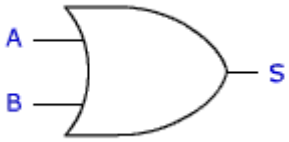


$S = \overline{A}$

A	S
0	1
1	0

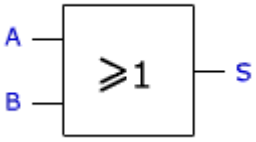


2/ porte OU

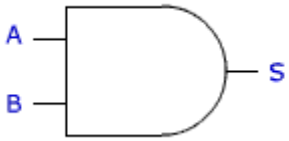


$S = A + B$

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

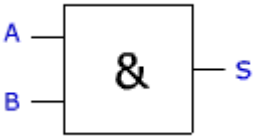


3/ porte ET

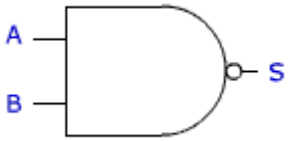


$S = A . B$

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

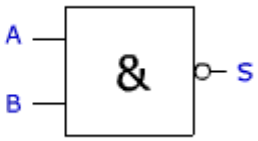


4/ porte NAND (NON ET)

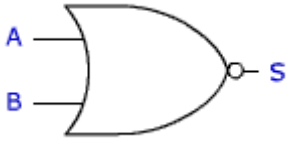


$S = \overline{A . B}$

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

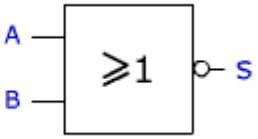


5/ porte NOR (NON OU)



$S = \overline{A + B}$

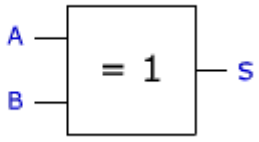
A	B	S
0	0	1
0	1	0
1	0	0
1	1	0



5/ porte XOR

$S = A \oplus B$

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0



$$S = \bar{A}B + A\bar{B}$$

Chapitre 3 : Recherche et simplification des fonctions logiques combinatoires

I-Introduction

Le fonctionnement d'un système logique combinatoire est décrit:

- Littéralement: par une ou plusieurs propositions logiques.

- Numériquement: par sa table de vérité (état de la sortie pour toute les combinaisons des variables d'entrées).

- Algébriquement: par une fonction logique (en associant les variables par les opérateurs ET, OU et NON.

- Par une table de fonctionnement: décomposition en plusieurs blocs fonctionnels.

II- Fonctions logiques décrites par une table de vérité:

1/ Fonction complétement définie:

Il s'agit de fonctions dont la valeur est connue pour toutes les combinaisons des variables.

-Exemple 1 : soit a,b et c les variables d'entrées et s la sortie.

a	b	c	s
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

2/ Fonction incomplètement définie:

Une fonction est dite incomplètement définie quand sa valeur est *indifférente* (ne change pas le résultat) ou *non spécifiée* (n'existent pas) pour certaines combinaisons de variables. Elles peuvent être physiquement impossibles. On utilise le symbole X ou ϕ pour la valeur non spécifiée de la fonction.

-Exemple 2 :

a	b	c	s
0	0	0	0
0	0	1	ϕ
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	ϕ
1	1	0	ϕ
1	1	1	1

3/ Recherche d'une fonction logique à partir de sa table de vérité :

Prenons le cas de l'exemple 1, S=1 pour les combinaisons N°2, 3,4 et 7

Donc $S = \bar{a}\,\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + ab\bar{c}$

III-Simplification des fonctions logiques

-Objectif : Fabriquer un système :

- à moindre coût
- rapide
- fiable
- peu consommateur

Deux méthodes de simplification sont utilisées:

- La réduction algébrique

- Les tableaux de KARNAUGH (diagramme de KARNAUGH).

1/ simplification algébrique

Il s'agit d'appliquer les principes et les propriétés de l'algèbre de Boole.

Prenons le cas de l'exemple 1 :

$$S = \bar{a}\,\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + ab\bar{c} = \bar{a}\,c\left(\bar{b} + b\right) + b\bar{c}(\bar{a} + a) = \bar{a}\,c + b\bar{c}$$

2/ simplification graphique par le tableau de KARNAUGH

La méthode de Karnaugh permet de visualiser une fonction et d'en tirer naturellement une écriture simplifiée. L'élément de base de cette méthode est la table de Karnaugh qui représente toutes les combinaisons d'états possibles pour un nombre de variables donné.

La table de Karnaugh est un outil graphique qui permet de simplifier de manière méthodique des expressions booléennes.

La construction des tables de Karnaugh exploite le codage de l'information et la notion d'adjacence (deux cases adjacentes leurs mots binaires ne diffèrent que par une, et une seule variable).

2.1-Règles de simplification

* Les groupements comportent une puissance de deux cases,

* Les 2^n cases forment un rectangle,

* On élimine le(s) variable(s) qui change(nt) d'état

-Groupement de 2^n cases élimine n variables

2 cases on élimine 1 variable;

4 cases on élimine 2 variables;

8 cases on élimine 3 variables;

* Il faut utiliser au moins une fois chaque 1, le résultat est donné par la réunion logique de chaque groupement

* Il faut grouper le maximum de cases et faire le minimum de groupement.

2.2- Diagrammes de Karnaugh

• Avec n = 2:

– Entrées A et B

– 4 cases

A / B	0	1
0		
1		

• Avec n = 3:

– Entrées A B et C

– 8 cases

AB / C	00	01	11	10
0				
1				

• Avec n = 4:

– Entrées A B C et D

– 16 cases

AB / CD	00	01	11	10
00				
01				
11				
10				

2.3- simplification des expressions logiques à l'aide du tableau de KARNAUGH

- Regroupement des cases adjacentes :
- Cas de 2 cases

Prenons le cas de l'exemple 1

S_1

ab / c	00	01	11	10
0	**0**	**1**	**1**	**0**
1	**1**	**1**	**0**	**0**

S_2

$S=S_1+S_2= \bar{a}\,c + b\bar{c}$

- Cas de 4 cases

$S = B\overline{C} + \overline{A}C$

AB / CD	00	01	11	10
00	**0**	**1**	**1**	**0**
01	**0**	**1**	**1**	**0**
11	**1**	**1**	**0**	**0**
10	**1**	**1**	**0**	**0**

- Cas d'une fonction incomplètement définie :
  Le symbole peut prendre indifféremment la valeur 0 ou 1; on remplace donc par 1 uniquement ceux qui permettent de simplifier une expression par regroupement.

$S = a + b\overline{c}$

ab / c	00	01	11	10
0	**0**	**1**	**ϕ**	**1**
1	**ϕ**	**0**	**1**	**ϕ**

PARTIE B

SYSTEME SEQUENTIEL

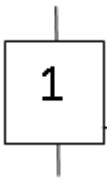
Chapitre 1 : LE GRAFCET

I-Eléments de base :

Le GRAFCET est utilisé dans la représentation graphique conventionnelle des systèmes automatisés. Il se compose d'un ensemble :

- d'**ETAPES** auxquelles sont associées des **ACTIONS**.
- de **TRANSITIONS** auxquelles sont associées des **RECEPTIVITEES**.
- de **LIAISONS ORIENTEES** reliant les étapes aux transitions et les transitions aux étapes.

1/ Etape et actions associées :



A un instant donné et suivant l'évolution du système :

- Une étape est soit ACTIVE soit INACTIVE ;
- L'ensemble des étapes active définit la situation de la partie commande.

L'étape initiale est représentée par un double carré.

Une ou plusieurs ACTIONS élémentaires peuvent être associées à une étape. Elles traduisent ce que l'on doit faire à chaque fois que l'étape à laquelle elles sont associées est active.

2/ Transition et réceptivité associées :

Une **TRANSITION** indique la possibilité d'évolution entre étapes. Cette évolution s'accomplit par le franchissement de la transition.

Une transition est soit **VALIDEE** soit **NON VALIDEE**

Elle est dite validée lorsque toutes les étapes précédentes immédiatement reliées à cette transition sont effectivement achevées.

A chaque transition est associée une proposition logique appelée **réceptivité** qui peut être soit **vraie** soit **fausse**.

II- Application :

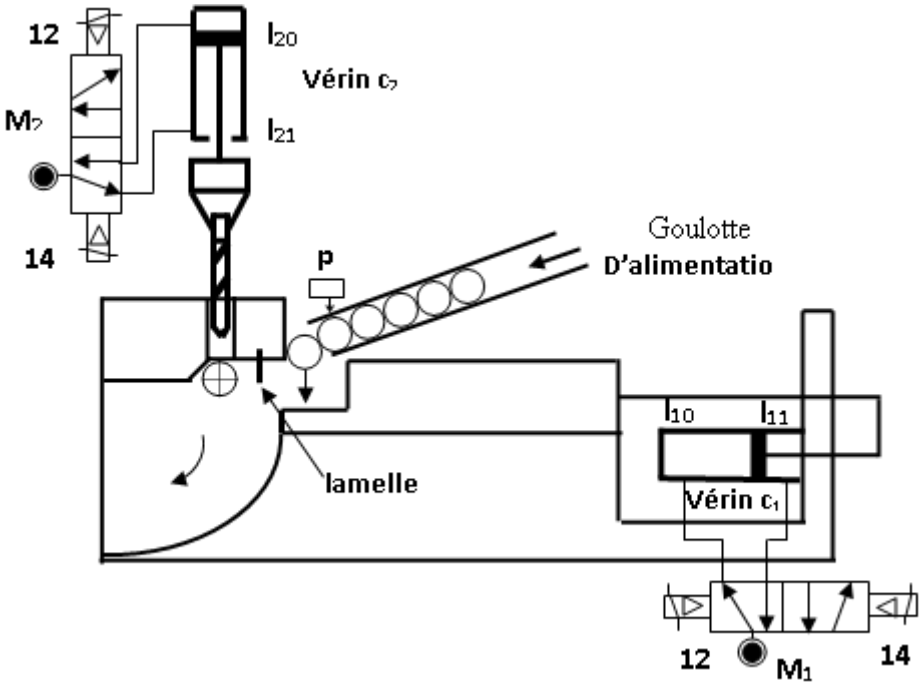
- **Poste de perçage :**

- **Description :**

Le poste de perçage est composé essentiellement, d'un système de serrage et de perçage. Les vérins C_1 et C_2 sont à double effet, pilotés par des distributeurs 5/2, bistables et à commande électropneumatique.

Les pièces à percer arrivent par gravitation.

- **Présentation :**



- **Fonctionnement :**

Les pièces arrivent par la goulotte d'alimentation ; un capteur p contrôle la présence des pièces et déclenche les opérations suivantes : serrage puis perçage.

Une lamelle empêche la pièce à percer de revenir dans le tiroir, elle tombe par basculement.

- **Travail demandé :**

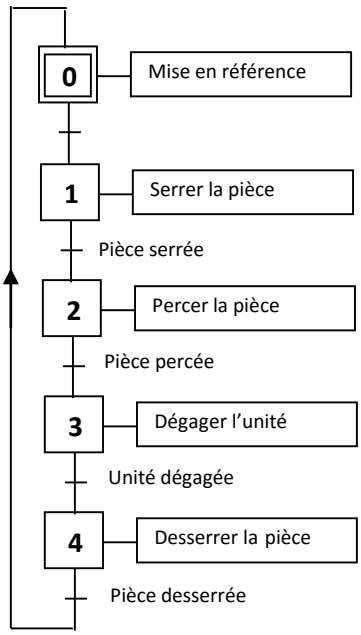
Décrire le fonctionnement du système par un grafcet selon les trois points de vue :

a-) **système** b-) **P.O** c-) **P.C**

- **Réponse :**

a - GRAFCET selon " un point de vue système " :

Ce grafcet correspond à un graphe de coordination des tâches décrivant le procédé de manière très générale sans présager des moyens techniques qui sont mis en œuvre pour le réaliser.



- **Règles d'évolution :**

- situation initiale (règle 1) : elle caractérise le comportement initiale de la P.C et correspond aux étapes actives au début de fonctionnement.

- franchissement d'une transition (règle 2) : une transition est franchie si toutes les étapes immédiatement précédentes sont actives et la réceptivité associée est vraie.

- Evolutions des étapes actives (règle 3) : le franchissement d'une transition entraîne l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes.

- Evolutions simultanées : (règle 4) : toutes les transitions franchissables sont simultanément franchies.

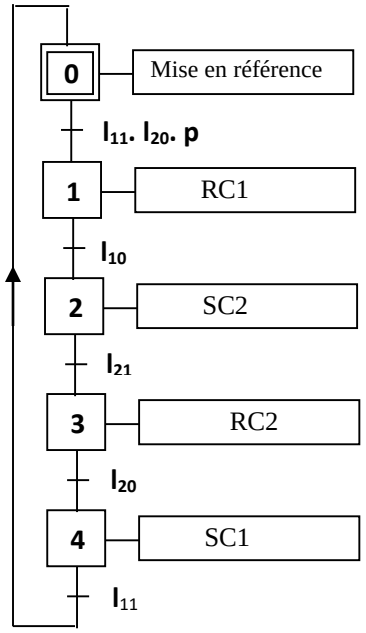
- Activations et désactivations simultanées (règle 5) : si, au cours du fonctionnement d'un automatisme, une même étape est en même temps désactivée et activée, elle reste active.

b - Grafcet PO :

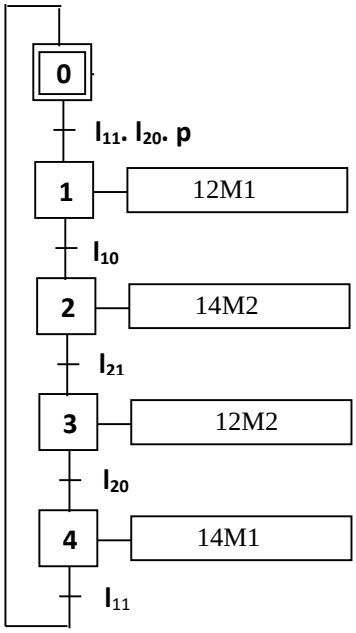
Ce Grafcet décrit le comportement de la partie commande pour obtenir des résultats souhaitables sur la partie opérative. On ne tient pas compte de la technologie qui sera retenue par les pré-actionneurs, les actionneurs et les capteurs.

c – Grafcet PC:

Ce type de Grafcet prend en compte les choix technologiques. Il est conçu selon le point de vue du réalisateur.



PO



PC

III- Structure de base d'un GRAFCET :

Le GRAFCET peut être : à **séquences uniques**, à **séquences simultanées** ou à **sélection de séquences**

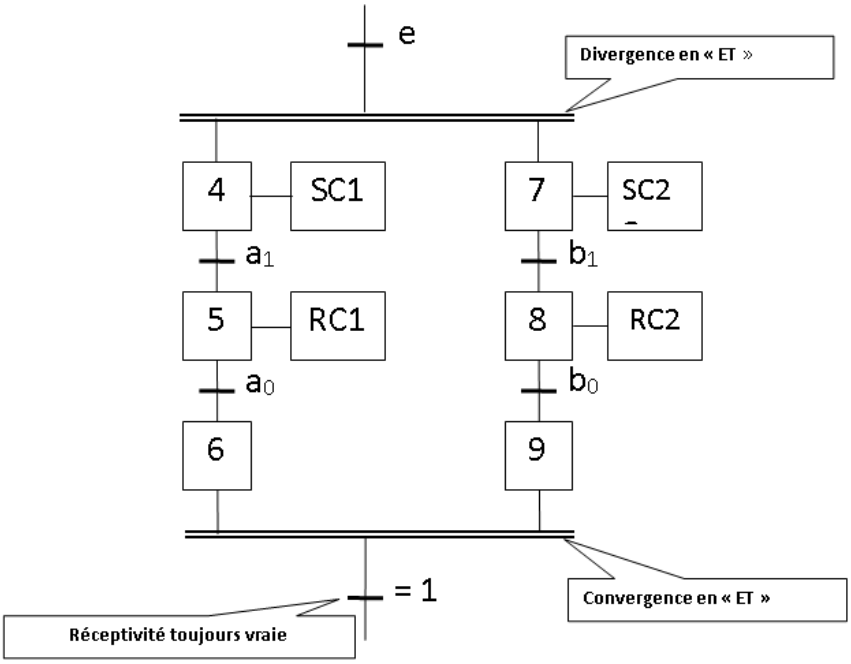
1/ Structure à séquences uniques :

Un grafcet linéaire est composé d'une suite d'étapes pouvant être activées les unes après les autres (application vu précédemment).

C'est le grafcet le plus souvent utilisé pour des systèmes simples.

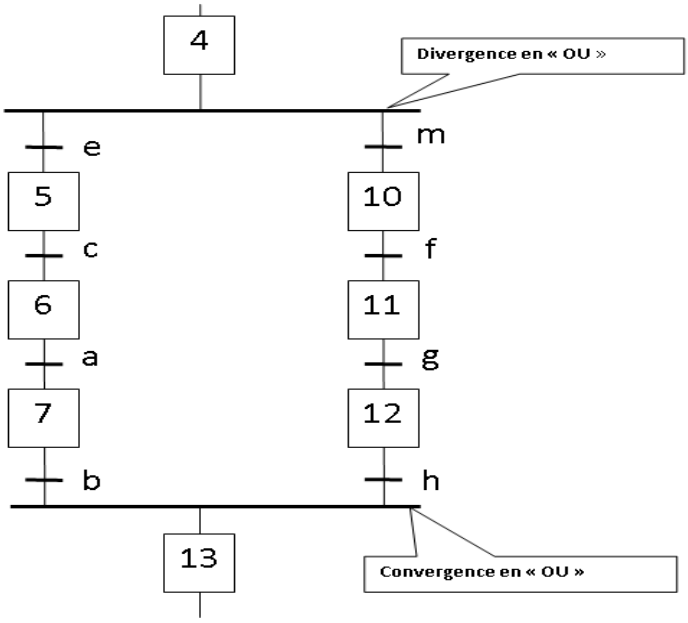
2/ Structure à séquences simultanées :

Les séquences sont dites simultanées lorsque plusieurs séquences sont activées en même temps, après le franchissement d'une même transition.

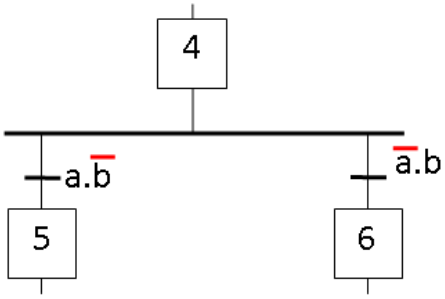


3/ Structure à sélection de séquences :

A partir d'une ou plusieurs étapes, il y a un choix d'évolution possible entre plusieurs étapes ou séquences.



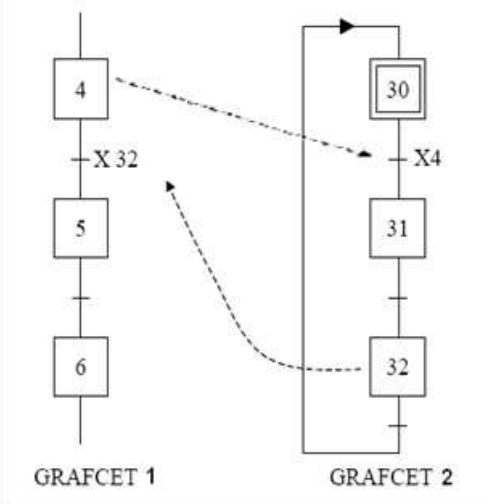
- Pour obtenir une sélection exclusive (OU exclusif), il faut s'assurer que les réceptivités associées aux transitions sont bien exclusives, c'est-à-dire non vraies simultanément.



Si a et b sont vraies en même temps, aucune transition ne pourra être franchie à partir de l'étape 4.

IV- Liaison entre GRAFCETS :

Une étape dans un grafcet peut servir comme réceptivité à une autre étape d'un autre grafcet. Cette méthode est utilisée aussi pour synchroniser deux grafcets c'est à dire rendre l'évolution de l'un dépend de l'évolution de l'autre.



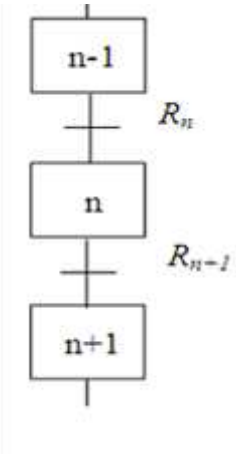
V- Mise en équation d'un GRAFCET :

1/ Règle générale :

Pour qu'une étape soit activée il faut que :

- L'étape immédiatement précédente soit active ;
- La réceptivité immédiatement précédente soit vraie ;
- L'étape immédiatement suivante soit non active ;

Après activation l'étape mémorise son état.

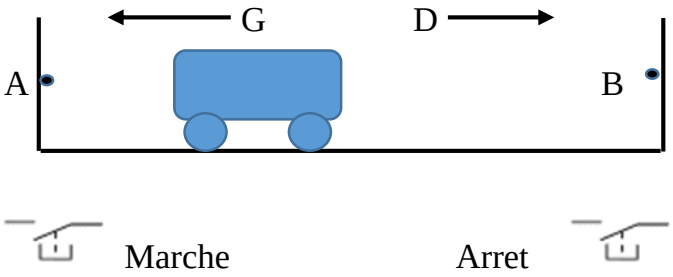


$$X_n = (X_{n-1}.R_n + X_n)\overline{X_{n+1}}$$

VI- Les notions avancés

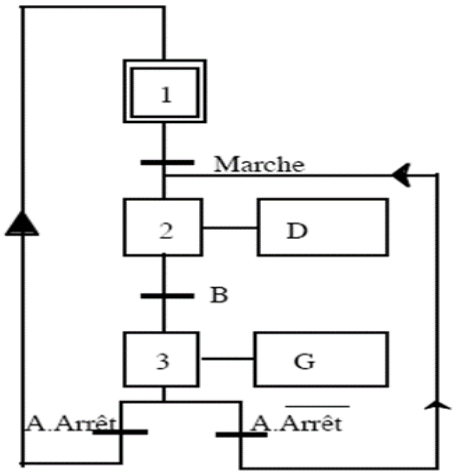
1/ Application

Soit le système automatisé suivant :

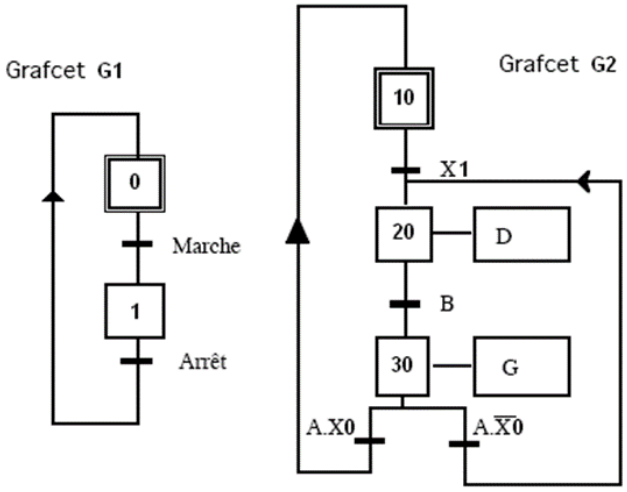


Le chariot fait des navettes entre un point A et un point B. On veut le commander grâce à deux boutons : Marche et Arrêt. C'est à dire que le chariot commence à effectuer ses cycles une fois le bouton Marche aura été appuyé et il arrêtera une fois son cycle terminé si l'on a appuyé sur le bouton Arrêt.

Proposition de grafcet :



Cette solution a un inconvénient : pour arrêter le chariot il faut appuyer sur Arrêt au moment où le chariot arrive en A. Ce n'est pas satisfaisant. L'idée est d'avoir un grafcet qui lit les boutons et qui commande l'autre. Ce sera un grafcet **superviseur** :



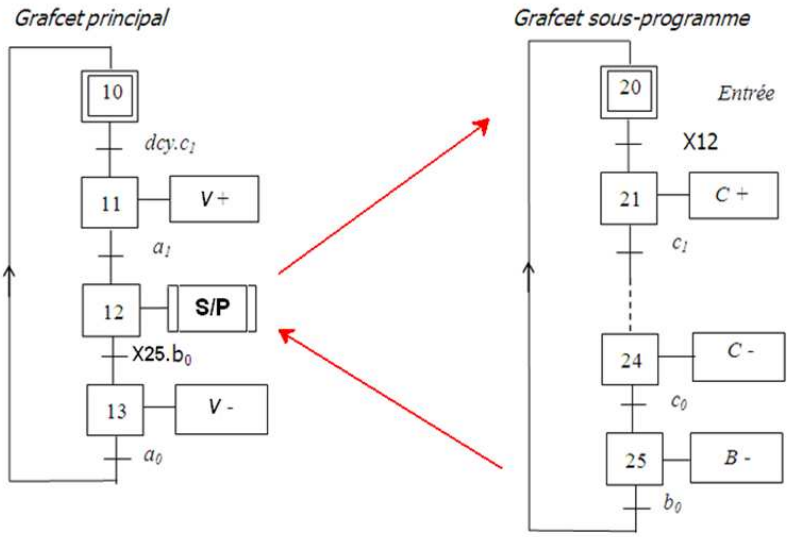
Dans les cas plus généraux et plus complexes, le grafcet maitre (superviseur) s'occupe du démarrage, de la séquence, de la synchronisation et de l'arrêt de différentes taches. Chaque tâche est décrite par un grafcet commandé par le superviseur ou grafcet de gestion des taches.

2/ Structure hiérarchisées d'un grafcet :

2.1- Structure d'un Sous-grafcet :

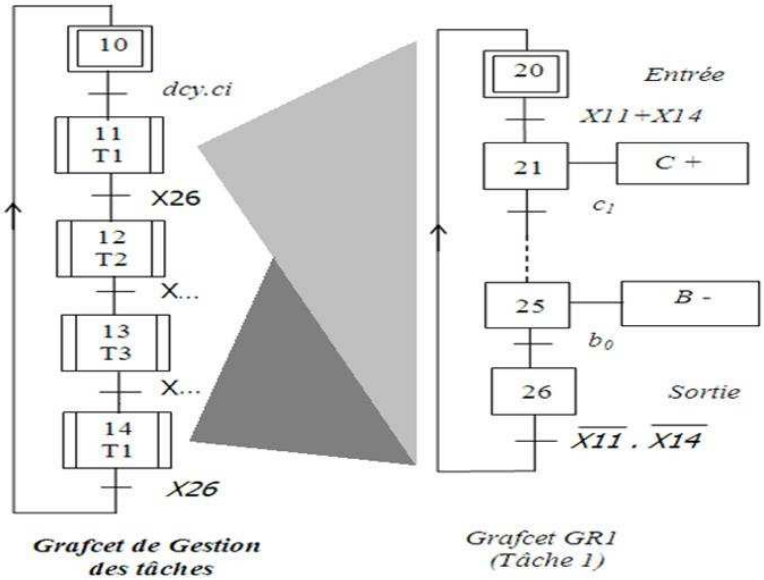
Dans les automatismes séquentiels, il est fréquent de rencontrer des séquences répétitives dans le même cycle. Une séquence répétitive peut être représentée par un sous-grafcet ou grafcet sous-programme.

Un grafcet sous-programme est écrit sous la forme d'un grafcet indépendant, connecté au grafcet principal.



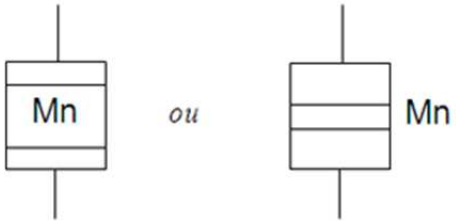
2.2- Structure d'un grafcet de tâche :

Les taches ont pour but de simplifier et de faciliter la description de systèmes complexes en allégeant le graphisme d'un grafcet et en détaillant séparément certaines parties.



2.3- Structure de l'expansion en macro-étape :

Une macro-étape est une représentation unique d'un ensemble d'étapes et de transitions appelée expansion de la macro-étape.



<u>Remarque</u> : Aucune action ne doit être associée à une macro-étape.

La structure de l'expansion obéit à certaines règles de construction à savoir :

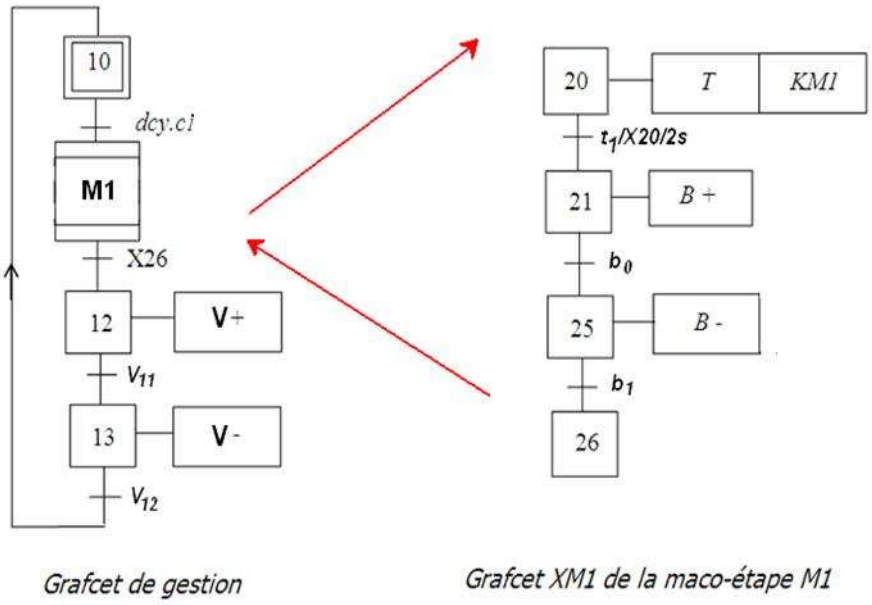
- l'expansion a toujours une étape d'entrée (E) et une étape de sortie (S) ;

- l'étape d'entrée est active dès le franchissement de la transition amont de la macro-étape ;

- l'activation de l'étape de sortie de l'expansion déclenche le franchissement de la transition aval de la macro-étape ;

- une expansion de macro-étape peut comporter des convergences en ET ou en OU, divergences en OU et en ET;

- une expansion de macro-étape peut contenir des macro-étapes.



Chapitre2 : LE GEMMA

I- Introduction

GEMMA : *Guide d'Etude des Modes de Marches et d'Arrêts*

Le GEMMA est un outil graphique qui permet de mieux définir les modes de marche et d'arrêt d'un système automatisé et de les prévoir dès la conception du système

Tout comme le GRAFCET, le GEMMA est un outil d'aide à l'analyse, il complète le GRAFCET.

II- Les états.

1/ Les états « F » *: Procédures de fonctionnement*

L'ensemble des modes ou états sans lesquels la valeur ajoutée ne peut être obtenue est regroupée dans une zone F représentative de la famille "procédure de fonctionnement".

Note : On ne produit pas systématiquement dans chacun des modes de cette famille, il peut s'agir de :

- Procédures préparatoire à la production de la valeur ajoutée

- réglages, tests ...

Ils sont néanmoins indispensables à la production de la valeur ajoutée.

- **F1 < Production normale >**

Dans cet état, la machine produit normalement : c'est l'état pour lequel elle a été conçue. C'est à ce titre que le "rectangle-état" a un cadre particulièrement renforcé. On peut souvent faire correspondre à cet état un GRAFCET que l'on appelle GRAFCET de base.

- **F2 < Marche de préparation >**

Cet état est utilisé pour les machines nécessitant une préparation préalable à la production normale : préchauffage de l'outillage, remplissage de la machine, mises en routes diverses, etc.

- **F3 < Marche de clôture >**

C'est l'état nécessaire pour certaines machines devant être vidées, nettoyées, etc., en fin de journée ou en fin de série.

- **F4 < Marche de vérification dans le désordre>**

Cet état permet de vérifier certaines fonctions ou certains mouvements sur la machine, sans respecter l'ordre du cycle.

- **F5 < Marche de vérification dans l'ordre >**

Dans cet état, le cycle de production peut être exploré au rythme voulu par la personne effectuant la vérification, la machine pouvant produire ou ne pas produire.

- **F6 < Marche de test >**

Les machines de contrôle, de mesure, de tri..., comportent des capteurs qui doivent être réglés ou étalonnés périodiquement : la < Marche de test > F6 permet ces opérations de réglage ou d'étalonnage.

2/ Les états « A » ***: Procédures d'arrêts.***

Un système automatisé ou une machine automatique fonctionne rarement de façon permanente : il s'avère nécessaire de l'arrêter de temps à autre pour des raisons indépendantes du système :

- Fin de journée,

- Période de congés,

- Manque d'approvisionnement ...

Tous les modes conduisant à (ou traduisant) un arrêt du système pour des raisons externes sont regroupés dans la zone A.

- **A1 <Arrêt dans l'état initial >**

C'est l'état "repos" de la machine. Il correspond en général à la situation initiale du GRAFCET : c'est pourquoi, comme une étape initiale, ce "rectangle-état" est entouré d'un double cadre.

Pour une étude plus facile de l'automatisme, il est recommandé de représenter la machine dans cet état initial.

- **A2 <Arrêt demandé en fin de cycle >**

Lorsque l'arrêt est demandé, la machine continue de produire jusqu'à la fin du cycle. A2 est donc un état transitoire vers l'état A1.

- **A3 < Arrêt demandé dans état déterminé >**

La machine continue de produire jusqu'à un arrêt en une position autre que la fin de cycle : c'est un état transitoire vers A4.

- **A4 < Arrêt obtenu >**

La machine est alors arrêtée en une autre position que la fin de cycle.

- **A5 < Préparation pour remise en route après défaillance >**

C'est dans cet état que l'on procède à toutes les opérations (dégagements, nettoyages,..) nécessaires à une remise en route après défaillance.

- **A6 < Mise P.O. dans état initial>**

La machine étant en A6, on remet manuellement ou automatiquement la Partie Opérative en position pour un redémarrage dans l'état initial.

„ **A7 < Mise P.O. dans état déterminé >**

La machine étant en A7, on remet la P.O. en position pour un redémarrage dans une position autre que l'état initial.

3/ Etats « D » ***: Procédures en défaillance***

Lors du fonctionnement d'un système, il peut se produire des incidents : on est donc conduit à prévoir les défaillances inhérentes ou internes au système.

Tous les modes conduisant (ou traduisant) un état d'arrêt du système pour des raisons internes sont consignés dans la zone D.

- **D1 <Arrêt d'urgence >**

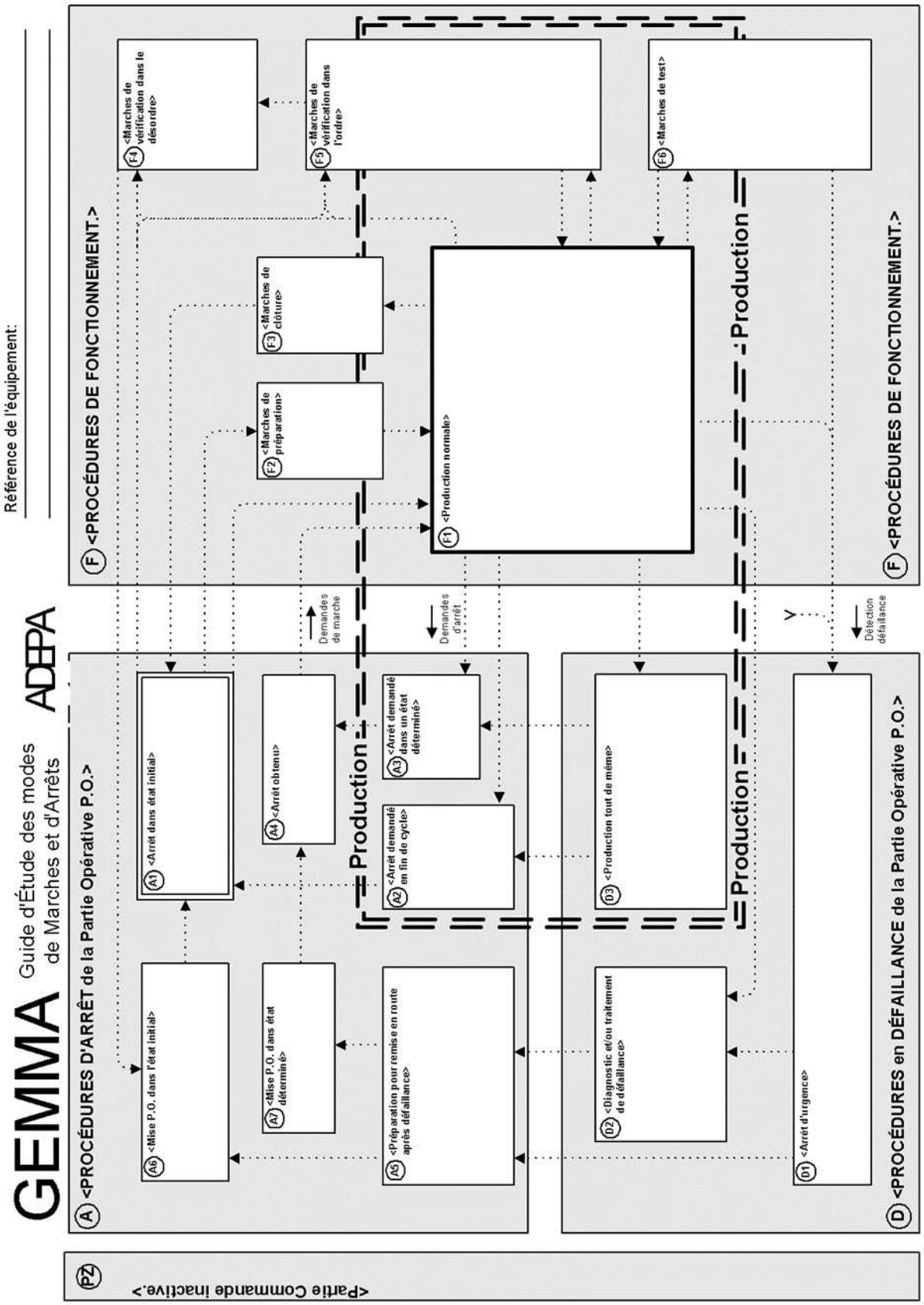
C'est l'état pris lors d'un arrêt d'urgence: on y prévoit non seulement les arrêts, mais aussi les cycles de dégagements, les procédures et précautions nécessaires pour éviter ou limiter les conséquences dues à la défaillance.

- **D2 < Diagnostic et/ou traitement de défaillance >**

C'est dans cet état que la machine peut être examinée après défaillance et qu'il peut être apporté un traitement permettant le redémarrage.

- **D3 < Production tout de même >**

Il est parfois nécessaire de continuer la production même après défaillance de la machine : on aura alors une "production dégradée", ou une "production forcée", ou une production aidée par des opérateurs non prévus en < Production normale >.


GEMMA
Guide d'Étude des modes de Marches et d'Arrêts
ADEPA
Référence de l'équipement:
A <PROCÉDURES D'ARRÊT de la Partie Opérative P.O.>
A1 <Arrêt dans état initial>
A6 <Mise P.O. dans l'état initial>
A4 <Arrêt obtenu>
A7 <Mise P.O. dans état déterminé>
Production
A2 <Arrêt demandé en fin de cycle>
A3 <Arrêt demandé dans un état déterminé>
A5 <Préparation pour remise en route après défaillance>
Demandes de marche
Demandes d'arrêt
F <PROCÉDURES DE FONCTIONNEMENT.>
F4 <Marches de vérification dans le désordre>
F3 <Marches de clôture>
F2 <Marches de préparation>
F1 <Production normale>
F5 <Marches de vérification dans l'ordre>
F6 <Marches de test>
Production
D <PROCÉDURES en DÉFAILLANCE de la Partie Opérative P.O.>
D3 <Production tout de même>
D2 <Diagnostic et/ou traitement de défaillance>
D1 <Arrêt d'urgence>
Production
Détection défaillance
PZ <Partie Commande inactive.>

III- Sélection des modes de marche et d'arrêt.

Il est nécessaire d'envisager l'ensemble des "rectangles-états" offerts par le GEMMA.

- Si le mode de fonctionnement est retenu, il doit être explicité conformément à la fonctionnalité de la machine.

- S'il n'est pas retenu dans le cadre de l'étude, une croix est portée dans le rectangle état.

Il est nécessaire de rechercher les évolutions d'un état à un autre sachant que dans tout système, on retrouve deux états fondamentaux :

- L'état A1, état initial ou repos.

- L'état F1, mode de production normale pour lequel le système a été conçu.

C'est à partir de ces deux états que sont recherchées les différentes évolutions.

Exemple :

- Le démarrage d'une machine, c'est à dire la transition de A1 vers F1 peut nécessiter une marche de préparation F2.

- L'arrêt peut se produire : Soit en fin de cycle F1--->A2---->A1 soit dans une autre position

F1--->A3---->A4

- Les cas de défaillances peuvent imposer : arrêt d'urgence D1 ou la production tout de même D3.

IV- Conditions d'évolution entre modes de marches et d'arrêts.

Les modes de marches et d'arrêt ayant été sélectionnés et explicités, il convient de préciser le passage d'un état à l'autre.

L'élaboration de ces conditions de passage rend possible **la conception du pupitre de commande** et entraîne éventuellement **l'adjonction de capteurs supplémentaires.**

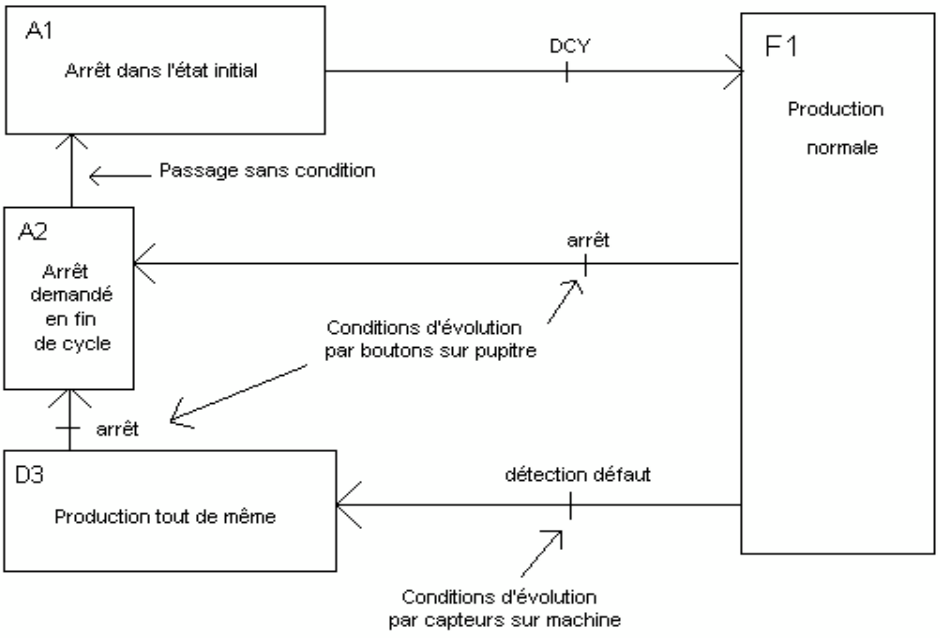
Le passage d'un état à un autre s'effectue de 2 façons :

- Soit avec une condition d'évolution.

- Soit sans condition d'évolution.

Avec les conditions d'évolution en provenance **de l'opérateur** apparaissent les besoins en **boutons de commande**.

A celles provenant de la **machine** doit correspondre la mise en place de **nouveaux capteurs**

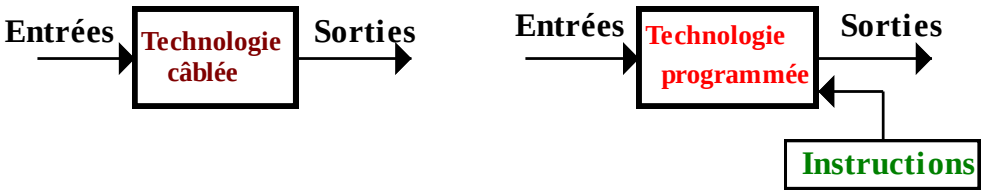


Chapitre 3 : Les automates programmables industriels (API)

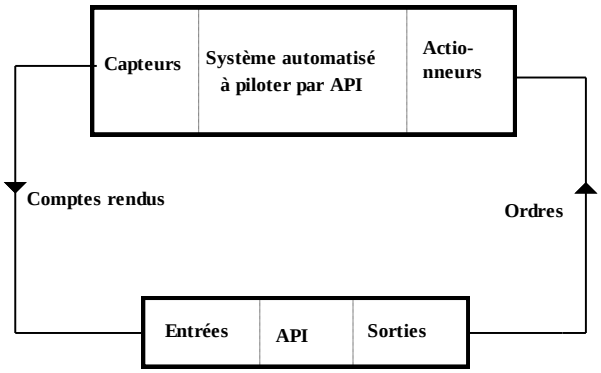
I- Introduction :

Les API sont des micro-ordinateurs simplifiés qui sont spécialement conçu pour traiter, par un programme, des problèmes de logique séquentielle (ou combinatoire) afin de remplacer les commandes d'automatismes en logique câblée réalisée avec des circuits logiques ou des relais. Ces équipements bien qu'ils offrent beaucoup moins de possibilités que les processus classiques , ont l'avantage d'être employés très facilement par des personnels non qualifiés en informatique , grâce à des langages de programmation de type langage à relais , équations booléennes ou GRAFCET . Les API sont en général conçues pour fonctionner en milieu industriel, afin de remplacer les armoires à relais.

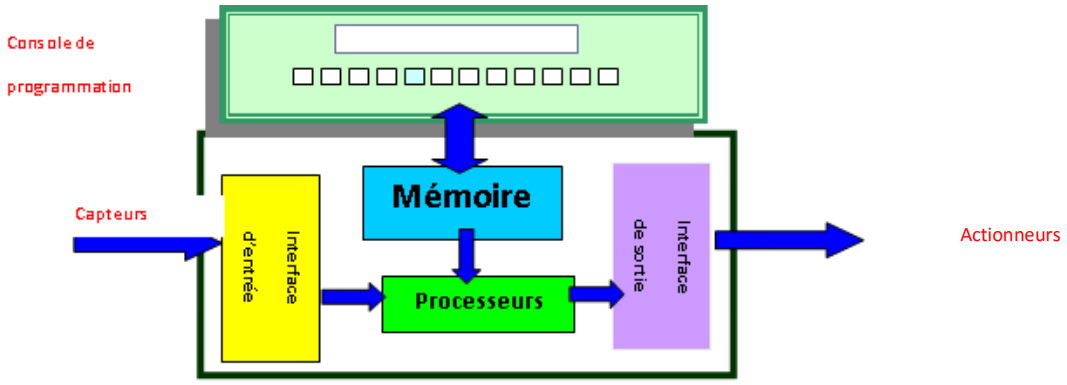
II- Structure générale d'une installation automatisée :



- Principaux constituants et connexions des API:



III- Structure générale des API



Les différents modules de l'API sont :

- Module d'entrées : Ils permettent à l'unité centrale de recevoir les informations logiques des capteurs et des boutons poussoirs du pupitre reliés à l'API.

- Module de sorties : ils permettent à l'unité centrale d'émettre des informations ordres de types logiques et électriques vers les pré-actionneurs de l'API.

- Unité centrale : l'unité centrale de traitement est l'ensemble électrique qui effectue le travail dans la mémoire programme. La mémoire programme enregistre les instructions du programme de commande ainsi que les données stockées à un endroit précis appelé " PAS" de programme.

Les trois parties sont reliées entre elles par des " BUS " (les bus sont des faisceaux de huit fils parallèles, chaque fil véhicule un bit : 0 ou 1).

- Module d'alimentation : deux alimentations de 24v continues ou alternatives pour assurer la liaison galvanique entre l'automate et le module d'entrées.

IV- Critères de choix d'un API :

Le choix d'un API est fonction de la partie commande à programmer. Choisir un API, ça revient à consulter les caractéristiques techniques suivantes :

- nombre d'E/S
- temps de traitement.
- nombre d'étapes ou instructions.
- nombre des temporisateurs.
- nombre des capteurs.
- capacité de la mémoire

V- Exemples d'API.

* Automates programmable OMRON :

- série CP1, CJ2

* Automates programmables TELEMECANIQUE :

- série TSX21 ; TSX17 ; TSX17-20 ; TSX27, 47, 67,87

* Automates programmables SCHNEIDER

- série TSX57, M340

* Automates programmables SIEMENS

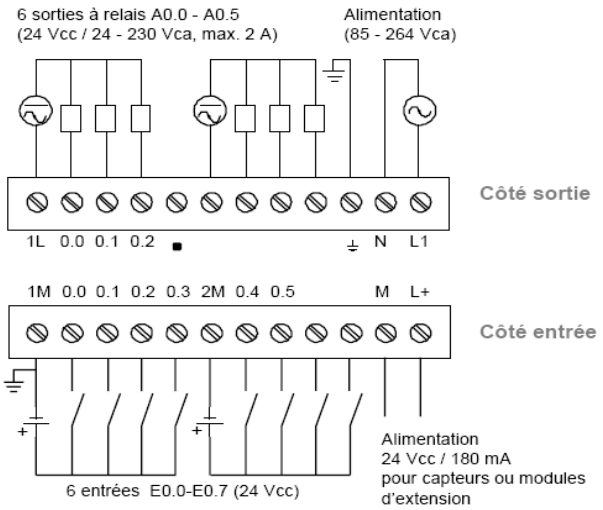
- série S7300, S7400, S71200, S71500

* Automates programmables MOELLER

- série PS4
- série XC

VI- Programmation de l'automate siemens

1/ Affectation des bornes du S7-200 (CPU 221)



2/ Types de variables

Il existe plusieurs types de variables, mais dans le cadre de cette première approche, on fera plus particulièrement attention aux variables suivantes:

- Entrées (E)
- Sorties (A)
- Mémoire (M)
- Temporisations (T)
- Compteurs (Z)

Les bits d'Entrées de l'automate sont notés :

E i . n *ou %I i.n*	Avec n = 0, 1, 2…7 le numéro du bit E = Eingang = entrée	Exemple : I0.0 : entrée 0 bit 0 (interrupteur)

Les bits de sortie de l'automate sont notés :

A i . n *ou %Q i.n*	Avec n = 0, 1, 2…7 le numéro du bit A = Ausgang = sortie	Exemple : Q0.0 : sortie 0 bit 0 (lampe)

Les bits internes de l'automate sont notés :

M i . n *ou %M i.n*	Avec n = 0, 1, 2…7 le numéro du bit M = Mémento = mémoire	Exemple : M10 : mémoire 10 (CF0)

- **Les temporisateurs (T):** le S7-200 fournit des temporisateurs qui comptent des incréments de temps selon des résolutions (incréments de base de temps) égales à 1ms, 10ms et 100ms.
- **Les compteurs (C)** : le S7-200 fournit trois types de compteurs qui comptent chaque transition du niveau bas au niveau haut aux entrées de comptage : les compteurs qui incrémentent, ceux qui décrémentent et ceux pouvant à la fois incrémenter et décrémenter.

3/ Jeu d'instructions :

Le processeur exécute un certain nombre d'opérations logiques; l'ensemble des instructions booléennes des instructions complémentaires de gestion de programme (saut, mémorisation, adressage ...) constitue un jeu d'instructions.

Chaque automate possède ses propres jeux d'instructions. Mais par contre, les constructeurs proposent tous une interface logicielle de programmation répondant à la norme CEI1131-3. Cette norme définit cinq langages de programmation utilisables, qui sont :

- *Les langages graphiques :*
  - **LD** : **L**adder **D**iagram (Diagrammes échelle)
  - **FBD** : **F**unction **B**lock **D**iagram (Logigrammes)
  - **SFC** : **S**equential **F**unction **C**hart (Grafcet)
- *Les langages textuels :*
  - **IL** : **I**nstruction **L**ist (Liste d'instructions).

- **ST**: **S**tructured **T**ext (Texte structuré).

Le langage à relais (Ladder Diagram) est basé sur un symbolisme très proche de celui utilisé pour les schémas de câblage classiques. Les symboles les plus utilisés sont donnés au tableau suivant :

Fonction	Symbole	
	Européen	Américain
Contact ouvert au ropos	---o o---	
Contact fermé au ropos	---o̅ o̅---	
Début de branchement		
Fin de branchement		
Affectation	---( )---	---( )

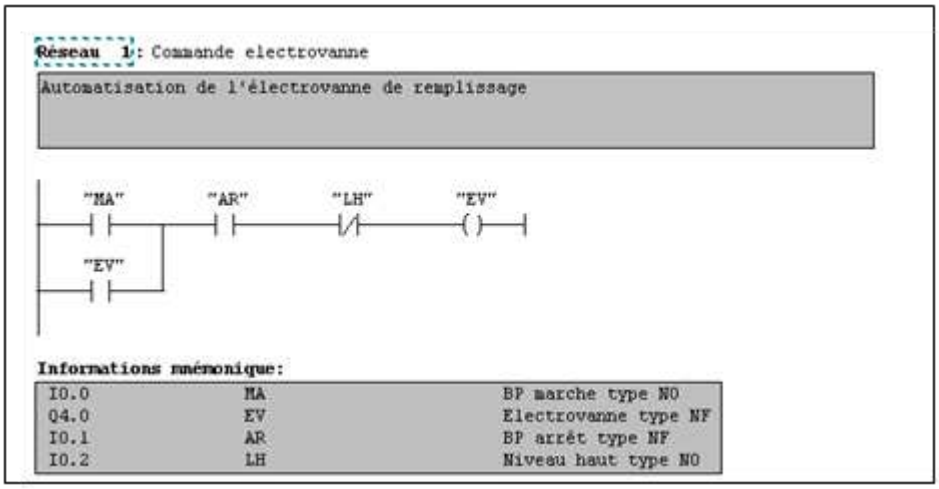
* Il existe plusieurs types de sorties pour l'API, mais nous n'allons en étudier que 3 sorties :

- Sortie directe : -( )- qui fonctionne comme l'entrée directe.

- Sortie SET (S) : -(S)- qui agit comme une mémoire et reste à 1.

- Sortie RESET (R) : -(R)- qui remet à zéro la sortie S.

- ***Exemple de programmation LADDER***



4/ Sécurité :

Les systèmes automatisés sont, par nature, source de nombreux dangers (tensions utilisées, déplacements mécaniques, jets de matière sous pression, etc.).

Placé au cœur du système automatisé, l'automate se doit d'être un élément fiable car un dysfonctionnement de celui-ci pourrait avoir de graves répercussions sur la sécurité des personnes, de plus les coûts de réparation et un arrêt de la production peuvent avoir de lourdes conséquences sur le plan financier.

Aussi, l'automate fait l'objet de nombreuses dispositions pour assurer la sécurité :

- **Contraintes extérieures** : l'automate est conçu pour supporter les différentes contraintes du monde industriel et a fait l'objet de nombreux tests normalisés.
- **Coupures d'alimentation** : l'automate est conçu pour supporter les coupures d'alimentation et permet, par programme, d'assurer un fonctionnement correct lors de la réalimentation (reprises à froid ou à chaud).
- **Mode RUN/STOP** : Seul un technicien peut mettre en marche ou arrêter un automate et la remise en marche se fait par une procédure d'initialisation (programmée)
- **Visualisation** : Les automates offrent un écran de visualisation où l'on peut voir l'évolution des entrées / sorties

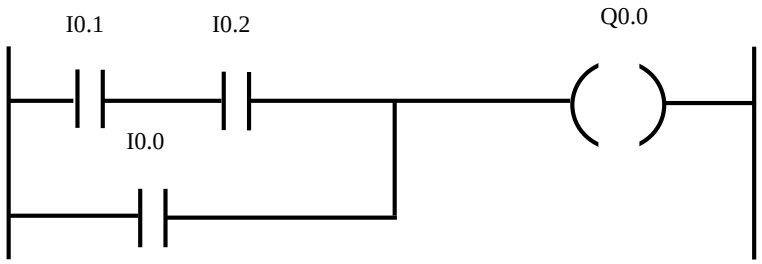
5/ Exemple de programmation LADDER :

Soit la fonction logique suivante : F = a + b.c

- Affectation des entrées et de la sortie

variables	affectation
a	I0.0
b	I0.1
c	I0.2
F	Q0.0

- Programmation LADDER



6/ Application :

- Poste de perçage (voir chapitre1 : GRAFCET)

  1) Ecrire les équations des étapes et des sorties

  2) Donner le programme LADDER